🧪 Visual Studio & CUDA Insights Debugging Demo (GPU-Level Profiling)

- **Kernel Execution Time per Batch**
- **Memory Bandwidth (Host ↔ Device)**
- **Streaming Multiprocessor Utilization**
- **Thread Block and Warp Efficiency**
- **Shared vs Global Memory Access Ratios**
- **Launch Configuration Visualizer**
- **Concurrent Execution / Stream Overlap Analysis**
- **Instruction Throughput / Arithmetic Intensity**
- **Synchronization Events and Bottlenecks**
- **FFT Plan Efficiency and Reuse Patterns**

📊 .json Signal Summary Report (System-Level Telemetry Metrics)

This set captures the properties you want to track and visualize—either via external tools or Python post-processing:

📌 Signal Stats per Chunk:

- Max Voltage
- Min Voltage
- Voltage Range (`max - min`)
- Mean Voltage
- Standard Deviation (Noise Level)
- Outlier Count
- Outlier Density (relative to sample count)

🔍 Frequency & Power (if FFT is applied):

- Dominant Frequency
- Spectral Centroid
- Spectral Spread (if needed)
- Power per Batch (sum of squared magnitudes)
- Frequency Response Matrix (for heatmap visualization)

📈 Distribution & Trend Metrics:

- Histogram Binning of All Values
- Rolling Statistics (optional)
- Batch Timestamp / Time Offset
- Chunk Index
- Anomaly Flags or Status (e.g., saturation, dropout)

🧭 System Architecture Overview

🧩 C++ Main App: `CudaBigData`

Responsibilities:

- Execute FFT batch processing
- Read input from `.txt` or `.dat` files
- Write telemetry data to memory-mapped file: `mmSignalReport`
- Optionally emit timing summary JSON

Key Features:

- Conditional compilation for telemetry (`ENABLE_SIGNAL_REPORT`)
- No in-process analysis or visualization
- Clean separation between compute and reporting

📦 Memory-Mapped File: `mmSignalReport`

Purpose: Persistent telemetry log for post-run analysis

Structure:

- Chunk entries with:
  - Timestamp
  - Chunk index
  - FFT parameters
  - Raw or transformed telemetry data
- Written sequentially during batch run
- Read-only during post-processing

🐍 Python Modules

🧩 `signal_report.py`

Responsibilities:

- Parse `mmSignalReport`
- Analyze each chunk:
  - Mean, stddev, min/max
  - FFT peak detection
  - Anomaly flags
- Emit structured JSON report per run
- Generate `matplot` visualizations

Output:

- `run_name_telemetry.json`
- `run_name_plot.png` or interactive plot

🧩 `benchmark.py`

Responsibilities:

- Load multiple JSON reports
- Compare:
  - Run durations
  - Data volumes
  - Signal metrics
- Generate comparison plots and summaries

Functions:

- `compare_run_durations(vruns)`
- `compare_telemetry(vruns)`

🧩 Workflow Summary

```
[CudaBigData]
 └── FFT batch
 └── writes to mmSignalReport
 └── emits timing JSON (optional)

[Python: signal_report.py]
 └── reads mmSignalReport
 └── analyzes chunks
 └── writes telemetry JSON
 └── generates plots

[Python: benchmark.py]
 └── loads multiple JSONs
 └── compares performance and signal quality
 └── visualizes results
```
